SPIDER-DAY02

1. CSV数据持久化

1.1 CSV持久化概述

```
【1】作用
   将爬取的数据存放到本地的csv文件中

【2】使用流程
    2.1> 打开csv文件
    2.2> 初始化写入对象
    2.3> 写入数据(参数为列表)

【3】示例代码
    import csv
    with open('sky.csv','w') as f:
        writer = csv.writer(f)
        writer.writerow([])
```

1.2 CSV示例代码

```
import csv
with open('test.csv','w') as f:
    writer = csv.writer(f)
    writer.writerow(['超哥哥','25'])
```

1.3 笔趣阁CSV持久化

```
"""
目标:
    笔趣阁玄幻小说数据持久化到CSV
思路:
    1. 在 __init__() 中打开csv文件，因为csv文件只需要打开和关闭1次即可
    2. 在数据处理函数中将所抓取的数据处理成列表，使用writerow()方法写入
    3. 数据抓取完成后关闭文件
"""
```

```python
import re
import requests
import time
import random
import csv

class NovelSpider:
    def __init__(self):
        self.url = 'https://www.biqukan.cc/fenlei1/{}.html'
        self.headers = {'User-Agent' : 'Mozilla/5.0 (Windows NT 10.0; WOW64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/86.0.4240.193 Safari/537.36'}
        # 定义csv相关变量
        self.f = open('novel.csv', 'w')
        self.writer = csv.writer(self.f)

    def get_html(self, url):
        html = requests.get(url=url, headers=self.headers).content.decode('gbk', 'ignore')

        self.refunc(html)

    def refunc(self, html):
        """正则解析函数"""
        regex = '<div class="caption">.*?<a href="(.*?)" title="(.*?)">.*?<small.*?>(.*?)</small>.*?>(.*?)</p>'
        novel_info_list = re.findall(regex, html, re.S)
        for one_novel_info_tuple in novel_info_list:
            item = {}
            item['title'] = one_novel_info_tuple[1].strip()
            item['href'] = one_novel_info_tuple[0].strip()
            item['author'] = one_novel_info_tuple[2].strip()
            item['comment'] = one_novel_info_tuple[3].strip()
            print(item)
            # 将数据存入csv文件
            item_li = [
                item['title'],
                item['href'],
                item['author'],
                item['comment'],
            ]
            self.writer.writerow(item_li)

    def crawl(self):
        for page in range(1, 6):
            page_url = self.url.format(page)
            self.get_html(url=page_url)
            time.sleep(random.randint(1, 2))

        # 数据抓取完成后关闭文件
        self.f.close()

if __name__ == '__main__':
    spider = NovelSpider()
    spider.crawl()
```

2. MongoDB数据持久化

2.1 MongoDB介绍

```
【1】MongoDB为非关系型数据库,基于key-value方式存储
【2】MongoDB基于磁盘存储,而Redis基于内存
【3】MongoDB数据类型单一,就是JSON文档
    MySQL数据类型:数值类型、字符类型、枚举类型、日期时间类型
    Redis数据类型:字符串、列表、哈希、集合、有序集合
    MongoDB数据类型: JSON文档
    # 此生铭记: MongoDB是基于磁盘存储的非关系型数据库,数据类型很单一,值就是JSON文档
【4】和MySQL对比
    MySQL:   库 - 表   - 表记录
    MongoDB: 库 - 集合 - 文档
【5】特性
    MongoDB无需提前建库建集合,直接使用即可,会自动创建
```

2.2 MongoDB常用命令

```
【1】进入命令行:  mongo
【2】查看所有库:  show dbs
【3】切换库:      use 库名
【4】查看库中集合:show collections  |  show tables
【5】查看集合文档:db.集合名.find().pretty()
【6】统计文档个数:db.集合名.count()
【7】删除集合:    db.集合名.drop()
【8】删除库:      db.dropDatabase()
```

2.3 与Python交互

- **pymongo模块**

```
【1】模块名: pymongo
          sudo pip3 install pymongo
【2】使用流程
    2.1》创建数据库连接对象
    2.2》创建库对象(库可以不存在)
    2.3》创建集合对象(集合可以不存在)
    2.4》在集合中插入文档
```

- **示例代码**

```
"""
库: noveldb
集合: novelset
文档: {'title':'花千骨', 'actor':'美丽的赵丽颖'}
"""
import pymongo

# 创建3个对象: 连接对象 库对象 集合对象
conn = pymongo.MongoClient(host='localhost', port=27017)
db = conn['noveldb']
myset = db['novelset']
# 插入文档
myset.insert_one({'title':'花千骨', 'actor':'美丽的赵丽颖'})
```

- **笔趣阁数据持久化**

```
"""
目标:
    笔趣阁玄幻小说数据持久化MongoDB
思路:
    1. __init__()中定义MongoDB相关变量
    2. 将抓取的数据处理成字典，利用insert_one()方法存入数据库
"""
import re
import requests
import time
import random
import pymongo

class NovelSpider:
    def __init__(self):
        self.url = 'https://www.biqukan.cc/fenlei1/{}.html'
        self.headers = {'User-Agent' : 'Mozilla/5.0 (Windows NT 10.0; WOW64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/86.0.4240.193 Safari/537.36'}
        # 定义MongoDB相关变量
        self.conn = pymongo.MongoClient('localhost', 27017)
        self.db = self.conn['noveldb']
        self.myset = self.db['novelset']

    def get_html(self, url):
        html = requests.get(url=url, headers=self.headers).content.decode('gbk', 'ignore')

        self.refunc(html)

    def refunc(self, html):
        """正则解析函数"""
        regex = '<div class="caption">.*?<a href="(.*?)" title="(.*?)">.*?<small.*?>(.*?)</small>.*?>(.*?)</p>'
        novel_info_list = re.findall(regex, html, re.S)
        for one_novel_info_tuple in novel_info_list:
            item = {}
            item['title'] = one_novel_info_tuple[1].strip()
            item['href'] = one_novel_info_tuple[0].strip()
            item['author'] = one_novel_info_tuple[2].strip()
```

```
            item['comment'] = one_novel_info_tuple[3].strip()
            print(item)
            # 将数据存入mongodb数据库
            self.myset.insert_one(item)

    def crawl(self):
        for page in range(1, 6):
            page_url = self.url.format(page)
            self.get_html(url=page_url)
            time.sleep(random.randint(1, 2))

if __name__ == '__main__':
    spider = NovelSpider()
    spider.crawl()
```

3. 笔趣阁多级页面爬虫

3.1 项目需求

```
【1】爬取地址
   https://www.biqukan.cc/fenlei1/1.html

【2】爬取目标
   '玄幻小说'分类下所有小说的：小说名称、链接、作者、描述、最新更新章节、最新更新章节链接

【3】爬取分析
   *********一级页面需抓取***********
       1、小说名称
       2、小说详情页链接
       3、小说作者
       4、小说描述

   *********二级页面需抓取***********
       1、最新更新的章节名称
       2、最新更新的章节链接
```

3.2 项目实现流程

```
【1】确认数据来源 - 响应内容中存在所抓取数据!!!
【2】找URL地址规律
    第1页: https://www.biqukan.cc/fenlei1/1.html
    第2页: https://www.biqukan.cc/fenlei1/2.html
    第n页: https://www.biqukan.cc/fenlei1/n.html
【3】 写正则表达式
    3.1》一级页面正则表达式
        '<div class="caption">.*?<a href="(.*?)" title="(.*?)">.*?<small.*?>(.*?)</small>.*?>(.*?)</p>'
    3.2》二级页面正则表达式
        '<dd class="col-md-4"><a href="(.*?)">(.*?)</a></dd>'
【4】代码实现
```

3.3 代码实现

```
"""
目标:
    笔趣阁玄幻小说数据抓取
思路:
    1. 确认数据来源 - 右键 查看网页源代码,搜索关键字
    2. 确认静态,观察URL地址规律
    3. 写正则表达式
    4. 写代码
"""
import re
import requests
import time
import random

class NovelSpider:
    def __init__(self):
        self.url = 'https://www.biqukan.cc/fenlei1/{}.html'
        self.headers = {'User-Agent' : 'Mozilla/5.0 (Windows NT 10.0; WOW64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/86.0.4240.193 Safari/537.36'}

    def get_html(self, url):
        """功能函数1: 获取html"""
        html = requests.get(url=url, headers=self.headers).content.decode('gbk', 'ignore')

        return html

    def refunc(self, regex, html):
        """功能函数2: 正则解析"""
        r_list = re.findall(regex, html, re.S)

        return r_list

    def crawl(self, first_url):
        """爬虫逻辑函数"""
        # 一级页面开始: 小说名称、链接、作者、描述
        first_html = self.get_html(url=first_url)
```

```
        first_regex = '<div class="caption">.*?<a href="(.*?)" title="(.*?)">.*?<small.*?>(.*?)</small>.*?>(.*?)</p>'
        novel_info_list = self.refunc(regex=first_regex, html=first_html)
        for one_novel_info_tuple in novel_info_list:
            item = {}
            item['title'] = one_novel_info_tuple[1].strip()
            item['href'] = one_novel_info_tuple[0].strip()
            item['author'] = one_novel_info_tuple[2].strip()
            item['comment'] = one_novel_info_tuple[3].strip()
            # 获取小说的最新章节名称、链接
            self.get_novel_data(item)

    def get_novel_data(self, item):
        """获取小说最新章节名称、链接"""
        second_html = self.get_html(url=item['href'])
        second_regex = '<dd class="col-md-4"><a href="(.*?)">(.*?)</a></dd>'
        chapter_list = self.refunc(regex=second_regex, html=second_html)
        for one_chapter_tuple in chapter_list:
            item['chapter_name'] = one_chapter_tuple[1].strip()
            item['chapter_href'] = one_chapter_tuple[0].strip()
            print(item)

    def run(self):
        for page in range(1, 2):
            page_url = self.url.format(page)
            self.crawl(first_url=page_url)
            time.sleep(random.randint(1, 2))

if __name__ == '__main__':
    spider = NovelSpider()
    spider.run()
```

3.4 练习

```
【1】将小说信息存入'MongoDB数据库'
【2】将小说信息存入'novel_info.csv文件'
【3】将小说信息存入'MySQL数据库'
    create database noveldb2 charset utf8;
    use noveldb2;
    create table novel_tab(
    novel_title varchar(200),
    novel_href varchar(300),
    novel_author varchar(200),
    novel_comment varchar(500),
    chapter_name varchar(200),
    chapter_href varchar(300)
    )charset=utf8;
```

4. 增量爬虫

4.1 增量爬虫概述

```
【1】引言
    当我们在浏览相关网页的时候会发现，某些网站定时会在原有网页数据的基础上更新一批数据，
    例：1. 某电影网站会实时更新一批最近热门的电影
        2. 小说网站会根据作者创作的进度实时更新最新的章节数据等等
    当我们在爬虫的过程中遇到时，我们是否需要只爬取网站中最近更新的数据，而不每次都做全量爬虫呢?

【2】概念
    通过爬虫程序监测某网站数据更新的情况，以便可以爬取到该网站更新出的新数据
```

4.2 增量爬虫实现

```
【1】原理
    1.1》在发送请求之前判断这个URL是不是之前爬取过
        适用场景：'不断有新页面出现的网站，比如说小说的新章节，每天的最新新闻等'
    1.2》在解析内容后判断这部分内容是不是之前爬取过
        适用场景：'页面内容会更新的网站'

【2】实现
    2.1》将爬取过程中产生的url进行存储，存储在redis的set中。当下次进行数据爬取时，首先对即将要发起的请求对应的url在存储的url的set中做判断，如果存在则不进行请求，否则才进行请求。
    2.2》对爬取到的网页内容进行唯一标识的制定，然后将该唯一表示存储至redis的set中。当下次爬取到网页数据的时候，在进行持久化存储之前，首先可以先判断该数据的唯一标识在redis的set中是否存在，在决定是否进行持久化存储
```

4.3 笔趣阁增量爬虫

```python
"""
增量爬虫实现步骤：
    1. 在__init__()中连接redis数据库
    2. md5加密的功能函数
    3. 抓取具体数据之前通过sadd的返回值做判断
        返回值为1：为新更新，说明之前没有抓取过
        返回值为0：无需抓取，之前已经抓取过
"""
import re
import requests
import time
import random
import redis
from hashlib import md5

class NovelSpider:
    def __init__(self):
        self.url = 'https://www.biqukan.cc/fenlei1/{}.html'
```

```
        self.headers = {'User-Agent' : 'Mozilla/5.0 (Windows NT 10.0; WOW64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/86.0.4240.193 Safari/537.36'}
        # 连接redis
        self.r = redis.Redis(host='localhost', port=6379, db=0)

    def get_html(self, url):
        """功能函数1：获取html"""
        html = requests.get(url=url, headers=self.headers).content.decode('gbk', 'ignore')

        return html

    def refunc(self, regex, html):
        """功能函数2：正则解析"""
        r_list = re.findall(regex, html, re.S)

        return r_list

    def md5_href(self, href):
        """功能函数3：生成指纹"""
        m = md5()
        m.update(href.encode())

        return m.hexdigest()

    def crawl(self, first_url):
        """爬虫逻辑函数"""
        # 一级页面开始：小说名称、链接、作者、描述
        first_html = self.get_html(url=first_url)
        first_regex = '<div class="caption">.*?<a href="(.*?)" title="(.*?)">.*?<small.*?>(.*?)</small>.*?>(.*?)</p>'
        novel_info_list = self.refunc(regex=first_regex, html=first_html)
        for one_novel_info_tuple in novel_info_list:
            item = {}
            item['title'] = one_novel_info_tuple[1].strip()
            item['href'] = one_novel_info_tuple[0].strip()
            item['author'] = one_novel_info_tuple[2].strip()
            item['comment'] = one_novel_info_tuple[3].strip()
            # 获取小说的最新章节名称、链接
            self.get_novel_data(item)

    def get_novel_data(self, item):
        """获取小说最新章节名称、链接"""
        second_html = self.get_html(url=item['href'])
        second_regex = '<dd class="col-md-4"><a href="(.*?)">(.*?)</a></dd>'
        chapter_list = self.refunc(regex=second_regex, html=second_html)
        for one_chapter_tuple in chapter_list:
            item['chapter_name'] = one_chapter_tuple[1].strip()
            item['chapter_href'] = one_chapter_tuple[0].strip()
            print(item)
            finger = self.md5_href(item['chapter_href'])
            if self.r.sadd('novel:spiders', finger) == 1:
                print('章节有更新，开始抓取... ...')
            else:
                print('章节未更新,跳过此章节')

    def run(self):
        for page in range(1, 2):
```

```
            page_url = self.url.format(page)
            self.crawl(first_url=page_url)
            time.sleep(random.randint(1, 2))

if __name__ == '__main__':
    spider = NovelSpider()
    spider.run()
```

5. Chrome浏览器插件

```
【1】在线安装
    1.1> 下载插件 - google访问助手
    1.2> 安装插件 - google访问助手：Chrome浏览器-设置-更多工具-扩展程序-开发者模式-拖拽(解压后的插件)
    1.3> 在线安装其他插件 - 打开google访问助手 - google应用商店 - 搜索插件 - 添加即可

【2】爬虫常用插件
    2.1》google-access-helper ：谷歌访问助手,可访问 谷歌应用商店
    2.2》Xpath Helper：轻松获取HTML元素的xPath路径
        打开/关闭：Ctrl + Shift + x
    2.3》JsonView：格式化输出json格式数据
    2.4》Proxy SwitchyOmega：Chrome浏览器中的代理管理扩展程序
```

6. xpath解析

6.1 xpath定义

```
XPath即为XML路径语言，它是一种用来确定XML文档中某部分位置的语言，同样适用于HTML文档的检索
```

6.2 匹配演示

```
"""
匹配猫眼电影top100：https://maoyan.com/board/4
"""
【1】查找所有的dd节点
    //dd
【2】获取所有电影的名称的a节点：所有class属性值为name的a节点
    //p[@class="name"]/a
【3】获取dl节点下第2个dd节点的电影节点
    //dl[@class="board-wrapper"]/dd[2]
【4】获取所有电影详情页链接：获取每个电影的a节点的href的属性值
    //p[@class="name"]/a/@href

【注意】
```

```
    1> 只要涉及到条件,加 [] :
        //dl[@class="xxx"]   //dl/dd[1]  //dl/dd[last()]

    2> 只要获取属性值,加 @  :
        //dl[@id="xxx"]   //p/a/@href
```

6.3 xpath语法

- **选取节点**

```
【1】// ： 从所有节点中查找（包括子节点和后代节点）
【2】@  ： 获取属性值
 2.1> 使用场景1（属性值作为条件）
      //div[@class="movie-item-info"]
 2.2> 使用场景2（直接获取属性值）
      //div[@class="movie-item-info"]/a/img/@src

【3】练习 - 猫眼电影top100
 3.1> 匹配电影名称
     //div[@class="movie-item-info"]/p[1]/a/@title
 3.2> 匹配电影主演
     //div[@class="movie-item-info"]/p[2]/text()
 3.3> 匹配上映时间
     //div[@class="movie-item-info"]/p[3]/text()
 3.4> 匹配电影链接
     //div[@class="movie-item-info"]/p[1]/a/@href
```

- **匹配多路径（或）**

```
xpath表达式1 | xpath表达式2 | xpath表达式3
```

- **常用函数**

```
【1】text() ： 获取节点的文本内容
    xpath表达式末尾不加 /text() :则得到的结果为节点对象
    xpath表达式末尾加 /text() 或者 /@href ： 则得到结果为字符串

【2】contains() ： 匹配属性值中包含某些字符串节点
    匹配class属性值中包含 'movie-item' 这个字符串的 div 节点
     //div[contains(@class,"movie-item")]
```

- **终极总结**

```
【1】列表中存放字符串：['', '', '', ...]
    xpath表达式的末尾为：/text() 、/@href  得到的列表中为'字符串'

【2】列表中存放节点对象
    其他剩余所有情况得到的列表中均为'节点对象'
    [<element dd at xxxa>,<element dd at xxxb>,<element dd at xxxc>]
    [<element div at xxxa>,<element div at xxxb>]
    [<element p at xxxa>,<element p at xxxb>,<element p at xxxc>]
```

- **课堂练习**

```
# www.guazi.com  点击 我要买车

【1】匹配瓜子二手车,所有汽车的链接 ：
    //ul[@class="carlist clearfix js-top"]/li/a/@href
【2】匹配瓜子二手车-汽车详情页中,汽车的
    2.1)名称：  //h1[@class="titlebox"]/text()
    2.2)里程：  //li[@class="two"]/span/text()
    2.3)排量：  //li[@class="three"]/span/text()
    2.4)变速箱：//li[@class="last"]/span/text()
    2.5)价格：//span[@class="price-num"]/text()
```

7. 今日作业

```
【1】正则抓取豆瓣图书top250书籍信息
    地址：https://book.douban.com/top250?icn=index-book250-all
    抓取目标：书籍名称、书籍信息、书籍评分、书籍评论人数、书籍描述

【2】使用xpath helper在页面中匹配豆瓣图书top250的信息，写出对应的xpath表达式
    书籍名称：
    书籍信息：
    书籍评分：
    评论人数：
    书籍描述：
【3】瓜子二手车综合练习
    3.1> 官网地址：https://www.guazi.com/
         点击：我要买车
         示例URL地址：https://www.guazi.com/bj/buy/o1/#bread
    3.2> 所抓数据
         一级页面(1个数据)：汽车详情页链接
         二级页面(1个数据)：汽车名称
                          (自己决定是否扩展:行驶里程、排量、变速箱、价格)
    3.3> 要求
         1、将所抓数据存入MySQL数据库
         2、将所抓数据存入MongoDB数据库
         3、将所抓数据存入CSV文件
         4、做成增量爬虫(新更新的汽车在前面)
    3.4> 提示
         1、多级页面抓取：定义功能函数
         2、瓜子二手车验证了User-Agent和Cookie
            'Cookie、User-Agent、URL地址必须都是自己浏览器中的'
         3、如何抓取Cookie和User-Agent
```

```
            打开浏览器,输入 www.guazi.com 回车,点击我要买车
            F12 - network - All , 刷新页面
            控制台中,找到最上面的网络数据包 buy/ 并点击
            右侧 Headers - Request Headers ,从中复制 Cookie和User-Agent

# 定义headers成如下,要定义自己的,不要使用笔记中这个
headers = {
    'User-Agent' : 'Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/88.0.4324.104 Safari/537.36',
    'Cookie' : 'uuid=80480517-5cc9-4687-dd22-1df01a0fd096; ganji_uuid=156193160017306500266 9; antipas=10188x3t20k7510809E4CiBw788; lg=1; clueSourceCode=%2A%2300; user_city_id=214; Hm_lvt_bf3ee5b290ce731c7a4ce7a617256354=1610337161,1610677550,1612237012,1612343727; sessionid=46fba556-4e07-45d2-a41a-26807c199c08; close_finance_popup=2021-02-03; cainfo=%7B%22ca_a%22%3A%22-%22%2C%22ca_b%22%3A%22-%22%2C%22ca_s%22%3A%22self%22%2C%22ca_n%22%3A%22self%22%2C%22ca_medium%22%3A%22-%22%2C%22ca_term%22%3A%22-%22%2C%22ca_content%22%3A%22-%22%2C%22ca_campaign%22%3A%22-%22%2C%22ca_kw%22%3A%22-%22%2C%22ca_i%22%3A%22-%22%2C%22scode%22%3A%22-%22%2C%22keyword%22%3A%22-%22%2C%22ca_keywordid%22%3A%22-%22%2C%22display_finance_flag%22%3A%22-%22%2C%22platform%22%3A%221%22%2C%22version%22%3A1%2C%22client_ab%22%3A%22-%22%2C%22guid%22%3A%2280480517-5cc9-4687-dd22-1df01a0fd096%22%2C%22ca_city%22%3A%22langfang%22%2C%22sessionid%22%3A%2246fba556-4e07-45d2-a41a-26807c199c08%22%7D; _gl_tracker=%7B%22ca_source%22%3A%22-%22%2C%22ca_name%22%3A%22-%22%2C%22ca_kw%22%3A%22-%22%2C%22ca_id%22%3A%22-%22%2C%22ca_s%22%3A%22self%22%2C%22ca_n%22%3A%22-%22%2C%22ca_i%22%3A%22-%22%2C%22sid%22%3A62229861303%7D; cityDomain=nc; lng_lat=116.91892_39.95635; gps_type=1; preTime=%7B%22last%22%3A1612344234%2C%22this%22%3A1609990531%2C%22pre%22%3A1609990531%7D; Hm_lpvt_bf3ee5b290ce731c7a4ce7a617256354=1612344235',
}



# 情况1:
<h1 class="titlebox">
        奥迪A6L 2020款 40 TFSI 豪华致雅型
                                                                <span class="labels">准新车</span>
                                        </h1>

# 情况2:
<h1 class="titlebox">
        丰田 汉兰达 2015款 2.0T 四驱豪华版 7座
                                                </h1>
# 正则表达式
<h1 class="titlebox">(.*?)(?:<span class="labels">|</h1>)
```